LISTING OF CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

1.   (Currently Amended) In a multithreaded computing environment, a method of processing computing tasks, comprising:

defining a plurality of worker threads, each thread capable of processing a task;

defining a plurality of task queues, each task queue capable of queuing a plurality of tasks;

associating each task queue with a single respective worker thread;

assigning a task to a task queue in an essentially random fashion, by:

selecting a task queue;

determining whether the selected task queue is in a non-empty state;

repeating the steps of selecting and determining until an empty task queue is found; and

placing the task in the empty task queue; and

from a worker thread, processing a task, wherein the task is located, during the act of processing, in from a task queue not associated with the thread.

2.   (Currently Amended) The method of claim 1 wherein assigning a task selecting comprises selecting an empty a task queue believed to be empty.

3.   (Canceled)

4.   (Previously Presented) The method of claim 1 further comprising, from a worker thread, processing a task from an associated task queue.

5.   (Canceled)

6.   (Currently Amended) In a multithreaded computing environment, a method of

processing computing threads, comprising:

defining a plurality of worker threads, each thread capable of processing a task;

defining a plurality of task queues, each task queue capable of queuing a plurality of tasks accessible by the worker threads;

associating each task queue with a single respective worker thread;

assigning a task to an assigned task queue, by:

selecting a task queue;

determining whether the selected task queue is in a non-empty state;

repeating the steps of selecting and determining until an empty task queue is found; and

placing the task in the empty task queue, the empty task queue as a result being designated the assigned task queue; and

in a worker thread not associated with the assigned task queue, processing the task, wherein the task is located, during the act of processing, in the assigned task queue.

7.   (Original) The method of claim 6 where assigning comprises selecting the assigned task queue based on an essentially random number.

8.   (Currently Amended) The method of claim 6 wherein ~~assigning~~selecting comprises selecting ~~an empty~~a task queue believed to be empty.

9.   (Canceled)

10.   (Currently Amended) In a multithreaded computing environment, a system for processing tasks, comprising:

a plurality of worker threads, each thread capable of processing a task;

a plurality of task queues, each task queue capable of queuing a plurality of tasks and each task queue associated with a single respective worker thread;

a task scheduler for assigning a task to a task queue in an essentially random

fashion, by:

selecting a task queue;

determining whether the selected task queue is in a non-empty state;

repeating the steps of selecting and determining until an empty task queue

is found; and

placing the task in the empty task queue; and

a worker thread processing a task, wherein the task is located, during the act of

processing, in from a task queue not associated with the thread.

11.   (Currently Amended) The system of claim 10 wherein the task scheduler selects

an empty a task queue believed to be empty for assigning the task.

12.   (Canceled)

13.   (Previously Presented) The system of claim 10 further comprising a worker

thread processing a task from an associated task queue.

14.   (Canceled)

15.   (Currently Amended) In a multithreaded computing environment, a system for

processing computing threads, comprising:

a plurality of worker threads, each thread capable of processing a task;

a plurality of task queues, each task queue capable of queuing a plurality of tasks

accessible by the worker threads and each task queue associated with a respective

worker thread;

a task scheduler for assigning a task to an assigned task queue, by:

selecting a task queue;

determining whether the selected task queue is in a non-empty state;

repeating the steps of selecting and determining until an empty task queue

is found; and

placing the task in the empty task queue, the empty task queue as a result being designated the assigned task queue; and

wherein the assigned task is processed by a thread not associated with the assigned task queue, wherein the task is located, during the act of processing, in the assigned task queue.

16.   (Previously Presented) The system of claim 15 wherein the task scheduler selects the assigned task queue based on an essentially random number.

17.   (Currently Amended) The system of claim 15 wherein the task scheduler selects an empty a task queue believed to be empty for assigning the task.

18.   (Canceled)

19.   (Currently Amended) An article of manufacturing, comprising:

   a computer-readable storage medium;

   a computer implemented program for processing computing tasks in a multithreaded computing environment embodied in the storage medium, the comprising instructions for:

        defining a plurality of worker threads, each thread capable of processing a task;

        defining a plurality of task queues, each task queue capable of queuing a plurality of tasks;

        associating each task queue with a single respective worker thread;

        assigning a task to a task queue in an essentially random fashion, by:

            selecting a task queue;

            determining whether the selected task queue is in a non-empty state;

            repeating the steps of selecting and determining until an empty task queue is found; and

placing the task in the empty task queue; and

processing, in a worker thread, a task, wherein the task is located, during the act of processing, in from a task queue not associated with the thread.

20.   (Currently Amended) The article of claim 19 wherein the instructions for assigningselecting a task comprise selecting an empty a task queue believed to be empty.

21.   (Canceled)

22.   (Previously Presented) The article of claim 19 further comprising instructions for processing, in a worker thread, a task from an associated task queue.

23.   (Canceled)

24.   (Currently Amended) An article of manufacture, comprising:

a computer-readable storage medium;

a computer-implemented program for processing computing threads, in a multithreaded computing environment embodied in the storage medium, the program comprising instructions for:

defining a plurality of worker threads, each thread capable of processing a task;

defining a plurality of task queues, each task queue capable of queuing a plurality of tasks accessible by the worker threads;

associating each task queue with a single respective worker thread;

assigning a task to an assigned task queue, by:

selecting a task queue;

determining whether the selected task queue is in a non-empty state;

repeating the steps of selecting and determining until an empty task

<u>queue is found; and</u>

<u>placing the task in the empty task queue, the empty task queue as a result being designated the assigned task queue</u>; and

in a worker thread not associated with the assigned task queue, processing the task<u>, wherein the task is located, during the act of processing, in the assigned task queue</u>.

25.   (Original) The article of claim 24 where the instructions for assigning comprise selecting the assigned task queue based on an essentially random number.

26.   (Currently Amended) The article of claim 24 wherein the instructions for ~~assigning~~<u>selecting</u> comprise selecting ~~an empty~~ <u>a</u> task queue<u> believed to be empty</u>.

27.   (Canceled)

28.   (Currently Amended) In a multithreaded computing environment, a system for processing computing tasks, comprising:

means for defining a plurality of worker threads, each thread capable of processing a task;

means for defining a plurality of task queues, each task queue capable of queuing a plurality of tasks;

means for associating each task queue with a <u>single </u>respective worker thread;

means for assigning a task to a task queue in an essentially random fashion<u>, by:</u>

<u>means for selecting a task queue;</u>

<u>means for determining whether the selected task queue is in a non-empty state;</u>

<u>means for repeating the steps of selecting and determining until an empty task queue is found; and</u>

<u>means for placing the task in the empty task queue</u>; and

from a worker thread, means for processing a task<u>, wherein the task is located,</u>

during the act of processing, in from a task queue not associated with the thread.

29.   (Currently Amended) The system of claim 28 wherein the means for ~~assigning a task~~selecting comprises means for selecting ~~an empty~~ a task queue believed to be empty.

30.   (Canceled)

31.   (Previously Presented) The system of claim 28 further comprising, from a worker thread, means for processing a task from an associated task queue.

32.   (Canceled)

33.   (Currently Amended) In a multithreaded computing environment, a method of processing computing tasks, comprising:

    defining a plurality of worker threads, each thread capable of processing a task;

    defining a plurality of task queues, each task queue capable of queuing a plurality of tasks;

    associating each task queue with a single respective worker thread;

    assigning a task to an empty task queue in an essentially random fashion, by:

        selecting a task queue;

        determining whether the selected task queue is in a non-empty state;

        repeating the steps of selecting and determining until an empty task queue is found; and

        placing the task in the empty task queue; and

    from a worker thread, processing a task, wherein the task is located, during the act of processing, in from a task queue not associated with the thread.

34.   (Currently Amended) The method of claim 33 wherein ~~assigning a task~~selecting comprises selecting ~~an empty~~ a task queue believed to be empty.

35.   (Canceled)

36.   (Previously Presented) The method of claim 33 further comprising, from a worker thread, processing a task from an associated task queue.

37.   (Currently Amended) In a multithreaded computing environment, a method of processing computing tasks, comprising:

   defining a plurality of worker threads, each thread capable of processing a task;

   defining a plurality of task queues, each task queue capable of queuing a plurality of tasks;

   associating each task queue with a single respective worker thread, an associated task queue capable of storing tasks assigned to an associated worker thread;

   assigning a task to a task queue in an essentially random fashion, comprising:

      using a random number generator to identify an initial task queue;

      upon determining that the initial task queue is ~~not empty~~in a non-empty state, searching ~~the~~other task queues for a~~n~~ empty task queue; and

      upon finding an empty task queue, storing the task in the empty task queue; and

   from a worker thread, processing a task, wherein the task is located, during the act of processing, in ~~from~~ a task queue not ~~the~~ associated ~~task queue~~with the worker thread.

38.   (Currently Amended) In a computer, a system for processing computing threads, comprising:

   a plurality of worker threads, each thread capable of processing a task;

   a plurality of task queues, each task queue capable of queuing a plurality of tasks accessible by the worker threads and each task queue associated with a single respective worker thread, the associated task queue capable of storing tasks assigned

to the associated worker thread;

a task scheduler for assigning a task to an assigned task queue in an essentially random fashion, the task scheduler using a random number generator to identify an initial task queue, ~~searching the other task queues for an empty a queue,~~ upon determining that the initial task queue is ~~not empty~~in a non-empty state, searching other task queues for an empty task queue, and upon finding an empty task queue, the task scheduler stores the task in the empty task queue, the empty task queue as a result being designated the assigned task queue, ~~the~~ a worker thread processing ~~a~~the task, wherein the task is located, during the act of processing, in ~~from~~ the assigned task queue, which is not associated ~~queue~~with the worker thread.

39.   (Currently Amended) In a multithreaded computing environment, a system for processing computing threads, comprising:

means for defining a plurality of worker threads, each thread capable of processing a task;

means for defining a plurality of task queues, each task queue capable of queuing a plurality of tasks accessible by the worker threads;

means for associating each task queue with a single respective worker thread;

means for assigning a task to an assigned task queue, by:

means for selecting a task queue;

means for determining whether the selected task queue is in a non-empty state;

means for repeating the steps of selecting and determining until an empty task queue is found; and

means for placing the task in the empty task queue, the empty task queue as a result being designated the assigned task queue; and

in a worker thread not associated with the assigned task queue, means for processing the task, wherein the task is located, during the act of processing, in the assigned task queue.

40.   (Currently Amended) In a multithreaded computing environment, a system for processing computing tasks, comprising:

means for defining a plurality of worker threads, each thread capable of processing a task;

means for defining a plurality of task queues, each task queue capable of queuing a plurality of tasks;

means for associating each task queue with a single respective worker thread;

means for assigning a task to an empty task queue in an essentially random fashion, by:

means for selecting a task queue;

means for determining whether the selected task queue is in a non-empty state;

means for repeating the steps of selecting and determining until an empty task queue is found; and

means for placing the task in the empty task queue; and

from a worker thread, means for processing a task, wherein the task is located, during the act of processing, in from a task queue not associated with the thread.

41.   (Currently Amended) In a multithreaded computing environment, a system for processing computing tasks, comprising:

means for defining a plurality of worker threads, each thread capable of processing a task;

means for defining a plurality of task queues, each task queue capable of queuing a plurality of tasks;

means for associating each task queue with a single respective worker thread, an associated task queue capable of storing tasks assigned to an associated worker thread;

means for assigning a task to a task queue in an essentially random fashion, comprising:

means for using a random number generator to identify an initial task

queue;

upon determining that the initial task queue is ~~not empty~~in a non-empty state, means for searching ~~the~~ other task queues for an empty task queue; and

upon finding an empty task queue, means for storing the task in the empty task queue; and

from a worker thread, means for processing a task, wherein the task is located, during the act of processing, in ~~from~~ a task queue not ~~the~~ associated ~~task queue~~with the worker thread.

42. (Currently Amended) The method of claim 37 wherein processing comprises:

from a worker thread, processing a task from a task queue ~~not~~ associated with the thread.

43. (Currently Amended) The system of claim 38 wherein a task scheduler comprises:

a task scheduler for assigning a task to ~~an assigned~~a task queue in an essentially random fashion, the task scheduler using a random number generator to identify an initial task queue, ~~searching the other task queues for an empty queue,~~ upon determining that the initial task queue is ~~not empty~~in a non-empty state, searching other task queues for an empty task queue, and upon finding an empty task queue, the task scheduler stores the task in the empty task queue, the worker thread processing a task ~~not associated with the~~ assigned to the associated task queue.

44. (Currently Amended) The system of claim 41 wherein means for processing comprises:

from a worker thread, means for processing a task from a task queue ~~not~~ associated with the worker thread.

45. (New) The method of claim 1 wherein determining whether the selected task queue is in a non-empty state comprises:

checking a task in the selected task queue to determine if the task is in the

process of being removed by a worker thread; and

      checking a task in the selected task queue to determine if the task is being acted upon by a worker thread.

46. (New) The method of claim 6 wherein determining whether the selected task queue is in a non-empty state comprises:

      checking a task in the selected task queue to determine if the task is in the process of being removed by a worker thread; and

      checking a task in the selected task queue to determine if the task is being acted upon by a worker thread.

47. (New) The method of claim 33 wherein determining whether the selected task queue is in a non-empty state comprises:

      checking a task in the selected task queue to determine if the task is in the process of being removed by a worker thread; and

      checking a task in the selected task queue to determine if the task is being acted upon by a worker thread.

48. (New) The method of claim 37 comprising:

      determining that the initial task queue is in a non-empty state by:

            checking a task in the initial task queue to determine if the task is in the process of being removed by a worker thread; and

            checking a task in the initial task queue to determine if the task is being acted upon by a worker thread.

49. (New) The system of claim 38 wherein the task scheduler comprises:

a task scheduler for assigning a task to an assigned task queue in an essentially random fashion, the task scheduler using a random number generator to identify an initial task queue, the task scheduler determining that the initial task queue is in a non-empty state by: checking a task in the initial task queue to determine if the task is in the

process of being removed by a worker thread, and checking a task in the initial task queue to determine if the task is being acted upon by a worker thread, upon determining that the initial task queue is in a non-empty state, searching other task queues for an empty task queue, and upon finding an empty task queue, the task scheduler stores the task in the empty task queue, the empty task queue as a result being designated the assigned task queue, a worker thread processing the task, wherein the task is located, during the act of processing, in the assigned task queue, which is not associated with the worker thread.

50. (New) The method of claim 1 wherein assigning a task to a task queue in an essentially random fashion comprises:

    selecting a task queue;

    determining whether the selected task queue is in a non-empty state;

    if the selected task queue is in a non-empty state, stealing a next task in sequence from the non-empty selected task queue by moving the task from the non-empty selected task queue to an associated queue that is empty, otherwise repeating the steps of selecting and determining until a selected task queue in a non-empty state is found.

51. (New) The method of claim 6 wherein assigning a task to an assigned task queue comprises:

    selecting a task queue;

    determining whether the selected task queue is in a non-empty state;

    if the selected task queue is in a non-empty state, stealing a next task in sequence from the non-empty selected task queue by moving the task from the non-empty selected task queue to an associated queue that is empty, otherwise repeating the steps of selecting and determining until a selected task queue in a non-empty state is found.

52. (New) The method of claim 33 wherein assigning a task to an empty task queue in

an essentially random fashion comprises:

      selecting a task queue;

      determining whether the selected task queue is in a non-empty state;

      if the selected task queue is in a non-empty state, stealing a next task in sequence from the non-empty selected task queue by moving the task from the non-empty selected task queue to an associated queue that is empty, otherwise repeating the steps of selecting and determining until a selected task queue in a non-empty state is found.

53. (New) The method of claim 37 wherein assigning a task to a task queue in an essentially random fashion comprises:

      using a random number generator to identify an initial task queue;

      determining whether the initial task queue is in a non-empty state;

      if the initial task queue is in a non-empty state, stealing a next task in sequence from the non-empty initial task queue by moving the task from the non-empty initial task queue to an associated queue that is empty, otherwise repeating the steps of using and determining until an initial task queue in a non-empty state is found.

54. (New) The system of claim 38 wherein the task scheduler comprises:

      a task scheduler for assigning a task to an assigned task queue in an essentially random fashion, the task scheduler using a random number generator to identify an initial task queue, the task scheduler determining whether the initial task queue is in a non-empty state, if the initial task queue is in a non-empty state, stealing a next task in sequence from the non-empty initial task queue by moving the task from the non-empty initial task queue to an associated queue that is empty, otherwise using and determining until an initial task queue in a non-empty state is found, the associated empty task queue as a result being designated the assigned task queue, a worker thread processing the task, wherein the task is located, during the act of processing, in the assigned task queue, which is not associated with the worker thread.